Runtime Analysis Tools

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27

Runtime analysis tools that detect memory violations are helpful in eliminating memory-related defects that can lead to heap-based vulnerabilities. To be effective, the tools must be used with a test suite that evaluates failure modes as well as planned user scenarios.

Development Context

Dynamic memory management.

Technology Context

C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C dynamic memory management functions such as `malloc()`, `calloc()`, and `free()` [ISO/IEC 99] are prone to programmer mistakes that can lead to vulnerabilities resulting from buffer overflow in the heap, writing to already freed memory, and freeing the same memory multiple times (e.g. double-free vulnerabilities).

Description

Runtime analysis tools that can detect memory violations are extremely helpful in eliminating memory-related defects that can lead to heap-based vulnerabilities. To be effective, the tools must be used with a test suite that evaluates failure modes as well as planned user scenarios.

BoundsChecker

Compuware's BoundsChecker is a run-time error detection and debugging tool for C++ developers. BoundsChecker provides feature such as deadlock detection, COM and .NET call reporting, memory and resource viewers and garbage collection notification.

Purify

1. daisy:268 (Plakosh, Daniel)

Two runtime analysis tools are Purify and PurifyPlus from IBM (formerly Rational). Purify performs memory corruption and memory leak detection functions and is available for both Windows and Linux platforms [IBM 04]. It detects when a program reads or writes freed memory or frees non-heap or unallocated memory and identifies writes beyond the bounds of an array.

Dmalloc Library

The debug memory allocation library (dmalloc) replaces the system's `malloc()`, `realloc()`, `calloc()`, `free()`, and other memory management functions to provide configurable, runtime debug facilities. These facilities include memory-leak tracking, fence-post write detection, file/line number reporting, and general logging of statistics [Watson 04].

The dmalloc library replaces the heap library calls normally found in system libraries with its own versions. When you make a call to `malloc()`, for example, you are calling dmalloc's version of the memory allocation function. When you allocate memory with these functions, the dmalloc library maintains debug information about your pointer, including where it was allocated, how much memory was requested, and when the call was made. This information can be verified when the pointer is freed or reallocated. The dmalloc library makes sure the pointer has not been corrupted when you reallocate or free a memory address.

Electric Fence

Electric Fence can detect buffer overflows or unallocated memory references. Electric Fence implements guard pages, using the virtual memory hardware of your computer to place an inaccessible memory page immediately after (or before, as the user defines) each memory allocation. When software reads or writes this inaccessible page, the hardware issues a segmentation fault, stopping the program at the offending instruction and thus making it easy to find the erroneous statement with your favorite debugger. In a similar manner, memory that has been released by `free()` is made inaccessible, and any code that touches it will get a segmentation fault.

Gnu Checker

Checker finds memory errors at runtime and warns you when the program reads an uninitialized variable or memory area or accesses an unallocated memory area [FSF 04].

The malloc library of Checker is a robust but slower version of malloc. Checker issues warnings when `free()` or `realloc()` is called with a pointer that does not reference a valid memory chunk, including chunks that have already been freed. Checker's malloc refrains from reusing a freed block immediately to catch accesses to the block shortly after it has been freed.

Checker implements a garbage detector that can be called by your program as it runs (e.g., by a debugger such as GDB) or when you exit it. The garbage detector displays all the memory leaks along with the functions that called malloc.

Valgrind

Valgrind allows you to profile and debug Linux/IA-32 executables [Valgrind 04]. The system consists of a core, which provides a synthetic IA-32 CPU in software, and a series of tools, each of which performs a debugging, profiling, or similar task. The architecture is modular so that new tools can be created easily and without disturbing the existing structure.

Valgrind is closely tied to details of the CPU, operating system, and (to a lesser extent) the compiler and basic C libraries. Valgrind is available on several Linux platforms and is licensed under the GNU

General Public License, version 2.

Insure++

Parasoft Insure++ is an automated runtime application testing tool that detects elusive errors such as memory corruption, memory leaks, memory allocation errors, variable initialization errors, variable definition conflicts, pointer errors, library errors, I/O errors, and logic errors [Parasoft 04].

During compilation, Insure++ reads and analyzes the source code to insert tests and analysis functions around each line. Insure++ builds a database of all program elements. In particular, Insure++ checks for the following categories of dynamic memory issues:

- reading from or writing to freed memory
- passing dangling pointers as arguments to functions or returning them from functions
- freeing the same memory chunk multiple times
- attempting to free statically allocated memory
- freeing stack memory (local variables)
- passing a pointer to `free()` that doesn't point to the beginning of a memory block
- calls to free with NULL or uninitialized pointers
- passing arguments of the wrong data type to `malloc()`, `calloc()`, `realloc()`, or `free()`

Application Verifier

Application Verifier helps you discover compatibility issues common to application code for Windows platforms and also additional feature to help identify security, memory and locking issues. The Page Heap utility (which used to be distributed with the Windows Application Compatibility Toolkit) is incorporated into Application Verifier's Detect Heap Corruptions test. It focuses on corruptions versus leaks and finds almost any detectable heap-related bug.

One advantage of Application Verifier's page heap test is that many errors can be detected as they occur. For example, an off-by-one byte error at the end of a dynamically allocated buffer might cause an instant access violation. For error categories that cannot be detected instantly, the error report is delayed until the block is freed.

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming Languages — C.* International Organization for Standardization, 1999.
[IBM 04]	IBM. *Rational PurifyPlus.* http://www-306.ibm.com/software/awdtools/purifyplus (2004).
[Watson 04]	Watson, Gray. *Dmalloc — Debug Malloc Library.* http://dmalloc.com (2004).
[FSF 04]	Free Software Foundation. *Checker.*

http://www.gnu.org/software/checker/checker.html (2004).

[Valgrind 04] Valgrind. *Valgrind Latest News.* http://valgrind.kde.org (2004).

[Parasoft 04] Parasoft. *Automating C/C++ Application Testing with Parasoft Insure++ (Insure++ Technical Papers).* http://www.parasoft.com/jsp/smallbusiness/tool_description.jsp? (2004).

Pearson Education, Inc. Copyright

Fields

Name	Value
Copyright Holder	Pearson Education

Fields

Name	Value
is-content-area-overview	false
Content Areas	Knowledge/Coding Practices
SDLC Relevance	Implementation
Workflow State	Publishable